SERVLETS - INTERNATIONALIZATION

Before we proceed, let me explain three important terms:

- **Internationalization (i18n):** This means enabling a web site to provide different versions of content translated into the visitor's language or nationality.
- **Localization (l10n):** This means adding resources to a web site to adapt it to a particular geographical or cultural region for example Hindi translation to a web site.
- **locale:** This is a particular cultural or geographical region. It is usually referred to as a language symbol followed by a country symbol which are separated by an underscore. For example "en_US" represents english locale for US.

There are number of items which should be taken care while building up a global website. This tutorial would not give you complete detail on this but it would give you a good example on how you can offer your web page in different languages to internet community by differentiating their location ie. locale.

A servlet can pickup appropriate version of the site based on the requester's locale and provide appropriate site version according to the local language, culture and requirements. Following is the method of request object which returns Locale object.

```
java.util.Locale request.getLocale()
```

Detecting Locale:

Following are the important locale methods which you can use to detect requester's location, language and of course locale. All the below methods display country name and language name set in requester's browser.

S.N.	Method & Description
1	**String getCountry()** This method returns the country/region code in upper case for this locale in ISO 3166 2-letter format.
2	**String getDisplayCountry()** This method returns a name for the locale's country that is appropriate for display to the user.
3	**String getLanguage()** This method returns the language code in lower case for this locale in ISO 639 format.
4	**String getDisplayLanguage()** This method returns a name for the locale's language that is appropriate for display to the user.
5	**String getISO3Country()** This method returns a three-letter abbreviation for this locale's country.
6	**String getISO3Language()** This method returns a three-letter abbreviation for this locale's language.

Example:

This example shows how you display a language and associated country for a request:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Locale;

public class GetLocale extends HttpServlet{

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
            throws ServletException, IOException
  {
      //Get the client's Locale
      Locale locale = request.getLocale();
      String language = locale.getLanguage();
      String country = locale.getCountry();

      // Set response content type
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      String title = "Detecting Locale";
      String docType =
      "<!doctype html public \"-//w3c//dtd html 4.0 " +
      "transitional//en\">\n";
      out.println(docType +
        "<html>\n" +
        "<head><title>" + title + "</title></head>\n" +
        "<body bgcolor=\"#f0f0f0\">\n" +
        "<h1 align=\"center\">" + language + "</h1>\n" +
        "<h2 align=\"center\">" + country + "</h2>\n" +
        "</body></html>");
  }
}
```

Languages Setting:

A servlet can output a page written in a Western European language such as English, Spanish, German, French, Italian, Dutch etc. Here it is important to set Content-Language header to display all the characters properly.

Second point is to display all the special characters using HTML entities, For example, "&#241;" represents "ñ", and "&#161;" represents "¡" as follows:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Locale;

public class DisplaySpanish extends HttpServlet{

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
            throws ServletException, IOException
  {
    // Set response content type
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    // Set spanish language code.
    response.setHeader("Content-Language", "es");

    String title = "En Espa&ntilde;ol";
    String docType =
     "<!doctype html public \"-//w3c//dtd html 4.0 " +
     "transitional//en\">\n";
     out.println(docType +
     "<html>\n" +
     "<head><title>" + title + "</title></head>\n" +
     "<body bgcolor=\"#f0f0f0\">\n" +
     "<h1>" + "En Espa&ntilde;ol:" + "</h1>\n" +
     "<h1>" + "&iexcl;Hola Mundo!" + "</h1>\n" +
     "</body></html>");
  }
}
```

Locale Specific Dates:

You can use the java.text.DateFormat class and its static getDateTimeInstance( ) method to format date and time specific to locale. Following is the example which shows how to format dates specific to a given locale:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Locale;
import java.text.DateFormat;
import java.util.Date;

public class DateLocale extends HttpServlet{

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
            throws ServletException, IOException
  {
    // Set response content type
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    //Get the client's Locale
    Locale locale = request.getLocale( );
    String date = DateFormat.getDateTimeInstance(
                                  DateFormat.FULL,
                                  DateFormat.SHORT,
                                  locale).format(new Date( ));

    String title = "Locale Specific Dates";
    String docType =
      "<!doctype html public \"-//w3c//dtd html 4.0 " +
      "transitional//en\">\n";
      out.println(docType +
      "<html>\n" +
      "<head><title>" + title + "</title></head>\n" +
      "<body bgcolor=\"#f0f0f0\">\n" +
      "<h1 align=\"center\">" + date + "</h1>\n" +
      "</body></html>");
  }
}
```

Locale Specific Currency

You can use the java.txt.NumberFormat class and its static getCurrencyInstance( ) method to format a number, such as a long or double type, in a locale specific curreny. Following is the example which shows how to format currency specific to a given locale:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Locale;
import java.text.NumberFormat;
import java.util.Date;

public class CurrencyLocale extends HttpServlet{

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
            throws ServletException, IOException
  {
    // Set response content type
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    //Get the client's Locale
    Locale locale = request.getLocale( );
    NumberFormat nft = NumberFormat.getCurrencyInstance(locale);
    String formattedCurr = nft.format(1000000);

    String title = "Locale Specific Currency";
    String docType =
      "<!doctype html public \"-//w3c//dtd html 4.0 " +
      "transitional//en\">\n";
      out.println(docType +
      "<html>\n" +
      "<head><title>" + title + "</title></head>\n" +
      "<body bgcolor=\"#f0f0f0\">\n" +
      "<h1 align=\"center\">" + formattedCurr + "</h1>\n" +
      "</body></html>");
  }
}
```

Locale Specific Percentage

You can use the java.txt.NumberFormat class and its static getPercentInstance( ) method to get locale specific percentage. Following is the example which shows how to format percentage specific to a given locale:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Locale;
import java.text.NumberFormat;
import java.util.Date;

public class PercentageLocale extends HttpServlet{

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
            throws ServletException, IOException
  {
    // Set response content type
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    //Get the client's Locale
    Locale locale = request.getLocale( );
    NumberFormat nft = NumberFormat.getPercentInstance(locale);
    String formattedPerc = nft.format(0.51);

    String title = "Locale Specific Percentage";
    String docType =
      "<!doctype html public \"-//w3c//dtd html 4.0 " +
      "transitional//en\">\n";
      out.println(docType +
      "<html>\n" +
      "<head><title>" + title + "</title></head>\n" +
      "<body bgcolor=\"#f0f0f0\">\n" +
      "<h1 align=\"center\">" + formattedPerc + "</h1>\n" +
      "</body></html>");
  }
}
```